Object Position Tracking with Convolution Neural Networks

Koushik Chennugari
CU Boulder
Boulder, CO 80309
koushik.chennugari@colorado.edu

Deepak Kancharla
CU Boulder
Boulder, CO 80309
surya.kancharla@colorado.edu

Rohit Mehra
CU Boulder
Boulder, CO 80309
rohit.mehra@colorado.edu

Nikhil Sulegaon
CU Boulder
Boulder, CO 80309
nisu8311@colorado.edu

Michael Walker
CU Boulder
Boulder, CO 80309
miwa2080@colorado.edu

Abstract

Augmented reality technology allows for interactions between virtual objects and stationary real objects, but it does not allow for interactions between virtual objects and moving real objects. One cannot place a hologram on a target person and/or robot and have it move appropriately with the target. Current solutions to this problem consist of using fiducial markers to estimate pose or the use of industrial grade motion capture cameras (Vicon, etc.) to capture the position of objects in a 3D environment. We propose a method that would allow us to track objects without the need for either fiducial markers and/or motion capture cameras, instead using convolutional neural networks. This method allows us to anchor holograms to dynamically moving objects using only images from a household webcam with a prediction accuracy of 97%.

1 Introduction

Attaching holograms to moving objects is a problem critical to the advancement of augmented reality (AR) technology. AR is a technology that overlays computer graphics onto real world environments in real-time. AR interfaces have three main features: (1) users are able to view both real and virtual objects in a combined scene, (2) users receive the impression that virtual objects are actually expressed and embedded directly in the real world, and (3) the virtual objects can be interacted with in real-time.

Using state-of-the-art AR head-mounted display (HMD) technology, such as that of the Microsoft HoloLens, spatial meshes of the user's environment can be created in real-time using spatial mapping techniques such as Kinect Fusion, which in turn allows the AR HMD to track its pose relative to the environment and for virtual objects to interact with surfaces in the user's real environment. This allows for functionalities such as dropping a virtual ball on a real desk and seeing it roll off the table and onto the floor or having a UI menu conform to the surface of a wall. However, this technique only works for stationary objects, which is a major limitation of current AR HMDs.

However, a solution to this limitation is likely to be found in the field of neural networks (NN). NN's have shown great success in the realm of computer vision. Tasks such as detecting whether or not an object is in an image or even classifying what object is in an image has been performed with great accuracy with NN's. Our team proposes a method of rapidly creating large, robust training datasets and a convolutional neural network (CNN) architecture for accurately predicting the 3D position (X, Y, Z) of a tracked object, using only images taken from a standard household webcam.

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

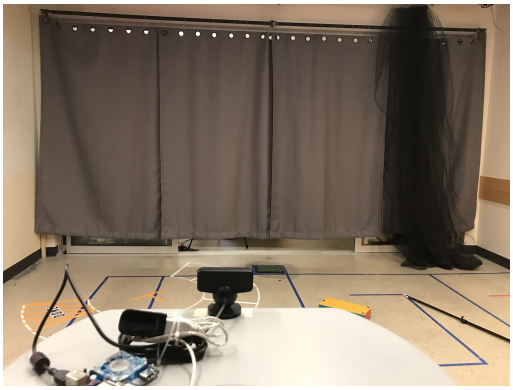

Figure 1: Lab environment and mounted webcam.

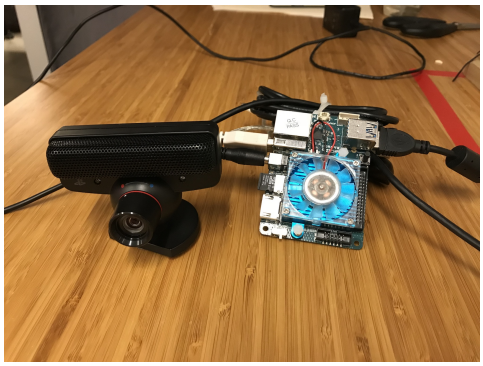

Figure 2: Webcam (Sony PLAYSTATION Eye) and Odroid (Heterogeneous Multi-Processing Octa-Core Linux Computer).

2 Data Collection

To create a dataset for our CNN, we needed to create a pipeline that allowed for the rapid creation of large, accurate datasets. This requirement was largely due to the fact that the features of the target tracked object (color, brightness, shape, rigidity, etc.) would impact the performance of the model, and multiple differing objects would need to be evaluated for tracking appropriateness.

To accomplish this, we used a network of Vicon motion capture cameras (measurements accurate to a millimeter) to track the 3D position of a target object in real-time. We utilized a 16ft x 16ft lab environment with unmarked solid-colored walls to decrease chance of environment visual feature interference such as object clutter and similar colors to our target object (see Figure 1). A webcam (Sony PLAYSTATION Eye) was mounted to a static location overlooking the lab environment and connected to an Odroid XU4 (Heterogeneous Multi-Processing Octa-Core Linux Computer) (see Figure 2). The object to be tracked was then attached to a 8ft boom and moved around the environment within the webcam's field of view (FOV). The webcam then took images of the object in the room at 30 fps. Whenever an image was taken of the object, the Odroid requested the current position of the object from the Vicon camera system, which was then paired with the capture image. A dataset providing an extremely accurate ground truth of over ten thousand labeled images could then be easily created in under ten minutes (see Figure 3).

3 CNN Design and Results

In this section, we present details of our CNN for object position estimation from a single image. Before we delineate the architectures of our model and how we arrived at it, we will elaborate on how model accuracy calculations were performed.

3.1 Calculating Accuracy

Our Deep CNN performs a regression task of predicting three values: X, Y, and Z coordinates of the object's position we are tracking. Due to the nature of the regression problem and the odds of predicting an estimation that is an exact match of any given label, we subjectively established a threshold for error classification. After performing manual testing with a Microsoft HoloLens AR HMD, our team chose 10 centimeters as our threshold. This threshold represents the minimum acceptable error for a hologram to be offset from its proper position (while overlaying a tracked object) and for the user to still associate the hologram with the target object. Therefore, a prediction is considered correct if and only if the absolute difference between the respective axes of the predicted and ground-truth coordinates of the object is less than 10 centimeters:

$$|\hat{y}_x - y_x| \leq 10 \; and \; |\hat{y}_y - y_y| \leq 10 \; and \; |\hat{y}_z - y_z| \leq 10$$

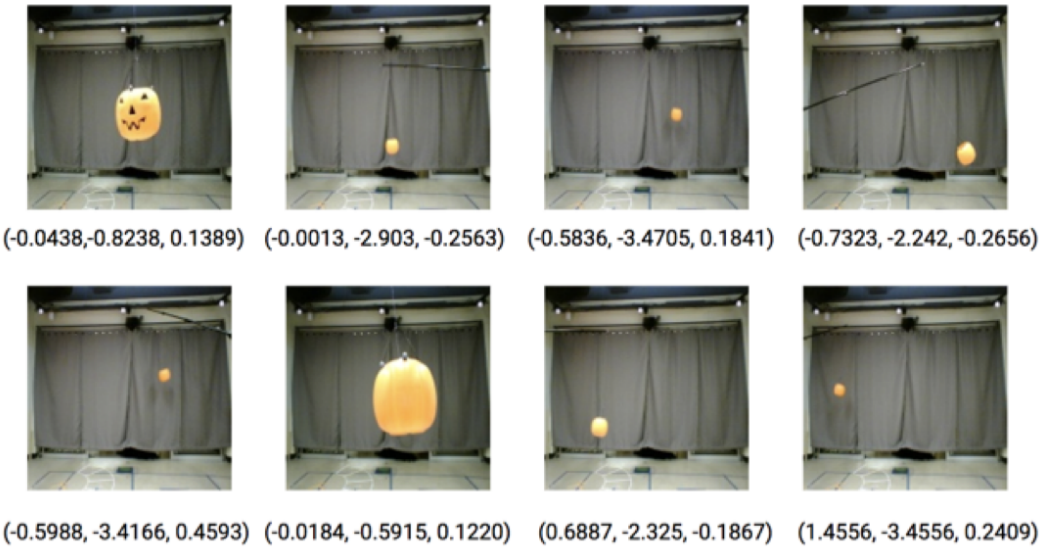

Figure 3: Sample dataset images with corresponding position labels(X, Y, Z).

Here, $\hat{y}_x$ and y_x are the predicted and ground-truth value of the x coordinate, $\hat{y}_y$ and y_y are the predicted and ground-truth value of the y coordinate, and $\hat{y}_z$ and y_z are the predicted and ground-truth value of the z coordinate respectively.

We then compute the accuracy(acc_{xyz}) of the model based on the total number of correct predictions. Apart from acc_{xyz}, we also evaluate a model based the individual accuracies in predicting the X, Y, and Z coordinates separately - acc_x, acc_y and acc_z. Splitting out these value's accuracies is critical for gaining insight toward how changes in the datasets features (camera lens, object appearance, camera position, environment visual features, lighting, etc.) impact the CNN's accuracy.

3.2 Model architecture

Our approach to building a CNN was to pipe together a group of convolutional blocks consisting of a convolutional layer and a pooling layer. One of our very initial models consisted of three convolutional blocks with three fully connected layers succeeding it as shown in Figure 4. Relu was used as the activation function after each of the convolutional layer. The loss function used was Mean Squared Error. This model gave an average accuracy(acc_{xyz}) of 42% with the individual accuracies being - acc_x = 78%, acc_y = 51% and acc_z = 89%.

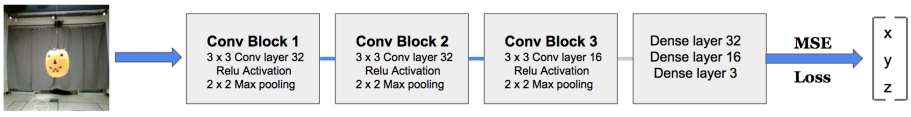

Figure 4: Interim Model 1.

To increase the accuracy further, we added three more Convolutional Blocks to the CNN as shown in Figure 5. As a consequence, the average accuracy of the model (acc_{xyz}) burgeoned to 83%. The average individual accuracies acc_x, acc_y, acc_z were 95%, 87% and 97% respectively.

We can infer from the observations of the above models that the model performance is being brought down by acc_y. We found the distortion caused by fish eye lens to be decreasing acc_y value predictions, and in turn, overall accuracy. Therefore, we changed the lens type of the camera as well as tweaked the CNN to use a different loss function to better predict the depth(acc_y) of the object from an image.

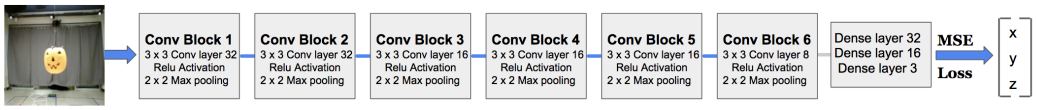

Figure 5: Interim Model 2.

3.3 Loss function

In our regression task, we used a straightforward loss function - L2 norm that minimizes the squared Euclidean distance between the estimated values ŷ and the ground-truth y. However, L2 norm loss function is not robust to outliers. Convex regression loss functions are considered not robust [1]. During back-propagation, small differences between the ground-truth and the estimated values have a little influence to the CNN weights modification, while large differences between the ground-truth and the estimated values (outliers) make a large penalty, which biases the whole training process towards the outliers [1]. Inspired by Robust Statistics [2, 3], we consider a non-convex loss function that is robust in regression tasks, namely Tukey's biweight loss function (Eq. 1) [2]. Tukey's biweight loss function has the property that the small residual values influence the training process, and it is robust to the outliers (large difference between the ground-truth depth value and the estimated depth value). During back-propagation, the magnitude of the outliers' gradient sets closed to zero, so it will suppress the influence of the outliers during the training process [3].

$$\rho(r) = \begin{cases} \frac{c^2}{6}\left[1 - \left(1 - \frac{r^2}{c^2}\right)^3\right], & |r| \leq c \\ \frac{c^2}{6}, & otherwise \end{cases} \tag{1}$$

Where c is a tuning constant that is usually set to 4.6851, and r is the residual (i.e. the difference between the ground-truth y and the estimated depth value ŷ).

$$r = \hat{y} - y \tag{2}$$

3.4 Effect of Tukey's Biweight Loss Function

Using a non-convex loss function like Tukey's biweight, pushed the accuracy of our model (acc_{xyz}) to 92% with the individual accuracies acc_x, acc_y, acc_z being 98%, 93%, and 99% respectively.

We also learned that regression tasks required fewer filters with small kernel size and almost no need to use fully connected layers compared to classification tasks. Thus, the number of fully connected layers in our CNN was reduced to one as shown in Figure 6. This not only extracted more powerful features from the input image, but also reduced the number of parameters in the network and its computational cost [3]. The accuracy(acc_{xyz}) with this model increased to 97% with individual accuracies acc_x, acc_y, acc_z being 98%, 98%, and 99% respectively. Table 1 provides a summary of the accuracies achieved with different models.

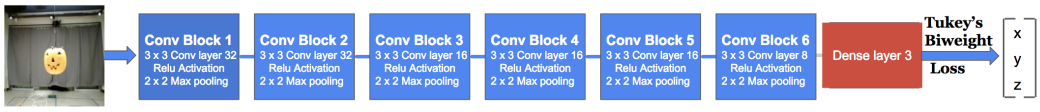

Figure 6: Final architecture of the CNN with Tukey's Biweight as loss function.

Table 1: Accuracies of different models when trained for 100 epochs

		Average Accuracy(5 trials)				
Model Architecture	Loss function	acc_{xyz}	acc_x	acc_y	acc_z	σ_{xyz}
3 Conv Blocks, 3 Dense	MSE	42%	78%	52%	89%	14%
6 Conv Blocks, 3 Dense	MSE	83%	95%	87%	97%	11%
6 Conv Blocks, 3 Dense	Tukey's Biweight	92%	98%	93%	99%	3%
6 Conv Blocks, 1 Dense	Tukey's Biweight	97%	98%	98%	99%	1%

4 Discussion

4.1 Object Selection

We started our data collection with a cuboid object with distinct colors on each of its sides. The prediction of acc_y becomes a concern in cases of asymmetric objects such as a rectangular cuboid, as the object may seem smaller or larger based on which object face is facing the camera even though it is in the same position in both instances. We designed our tracked object as such (rectangular cuboid) with each face colored differently to provide the CNN the ability to infer full pose (each side being unique provides enough information to estimate object rotation). Although we predicted this object design to perform well, we were obtaining only unacceptable accuracies of less than 10%. While reviewing our dataset, we noticed that the camera was not able to capture all the object's colors properly due to glare on the sides of object from inadequate lighting and the low camera quality. With this knowledge in mind, we decided to design a simpler object for data collection as well as reducing our model to only look at object position (X, Y, Z) and not its full pose (X, Y, Z, R, P, Q).

We then chose a spherical object with a bright single color as our test subject and collected data in a contained environment. We were able to achieve an accuracy of 97% by adding more convolution layers to aid in capture the object's curves. We achieved this accuracy after increasing our dataset from 6000 images to 9000 images.

5 Conclusion

5.1 Limitations and Future Work

Although we were able to achieve an accuracy above 95%, our solution to object tracking has many limitations. First of all, the current model is trained solely in a controlled lab environment. Our model would generalize poorly to environments outside of the lab due to other objects in the scene of similar size and shape as well as the environment backdrop having various confounding colors. Another limitation of this approach is that our model only predicts the relative position (X,Y,Z) of the target object and excludes rotational coordinates (roll, pitch, yaw). To properly anchor holograms to real objects one would need to know rotational information as well using camera image input and previous pose estimates, which even allows for moderate occlusion [4]. To make our model robust to occlusions in future work, we can augment the training data by occluding the object with another virtual object in a preprocessing step applied to a portion of the training samples [4].

The error threshold definition was a purely subjective decision and is a limitation within our experiment. Ideally we would run a separate experimental user study in which we would evaluate what the minimum acceptable amount of hologram offset is while still maintaining an acceptable level of usability and immersion. However, due to the scope of the class and time limitations, we were not able to explore this question further and made a judgment call using our intuition and first hand experience with AR HMDs. We recommend future studies explore this problem as it is essential for improving user experience with AR HMDs. The end goal of this future work is that of reducing the current slack value (10cm) and increasing accuracy. Ideally an accuracy equivalent to that of motion capture cameras (millimeters) would be targeted.

Images that would be obtained outside the lab environment would have a lot of features other than that of the target object. Future work should look at preprocessing the images to reduce or minimize irrelevant features prior to being passed into the NN for position prediction. Alternatively, future

work might explore the scalability of our designs in supporting the tracking of multiple objects and not only one as in this study.

5.2 Project Media

For a further look at our demo videos and data collection process please visit our project media repository: `https://drive.google.com/drive/folders/1iitwG0M6WlqesChOWIUcK1qrPzFrHt7g?usp=sharing`

References

[1] V. Belagiannis, C. Rupprecht, G. Carneiro, and N. Navab, “Robust optimization for deep regression,” in Proc. Conf. International Conference on Computer Vision, 2015.

[2] M. J. Black and A. Rangarajan, “On the unification of line processes, outlier rejection, and robust statistics with applications in early vision,” IJCV, 1996.

[3] A. J. Afifi and O. Hellwich, "Object Depth Estimation from a Single Image Using Fully Convolutional Neural Network," 2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA), Gold Coast, QLD, 2016, pp. 1-7.

[4] Garon, Mathieu, and Jean-François Lalonde. "Deep 6-DOF Tracking." arXiv preprint arXiv:1703.09771 (2017).